Методичка 8.1 - Знакомство с архитектурой x64

Основные преимущества архитектуры x64

Данная архитектура представляет собой расширение архитектуры x86 с полной обратной совместимостью. Существует множество вариантов названия данной архитектуры: AMD64, x86-64, x64 и можно еще найти несколько. В рамках этого курса мы будем называть ее просто x64.

Полная документация по данной архитектуре состоит из нескольких томов, мы же рассмотрим базовые моменты, которые нам будут полезны в рамках этого курса.

Основные преимущества архитектуры x64 перед x86:

- 64-битное адресное пространство
- расширенный набор регистров
- обратная совместимость с x86

Адресное пространство

Адресное пространство, по сравнению с архитектурой x86, стало сильно больше. 64-битный процессор теоретически может адресовать 16 экзабайт памяти (это 2^64). Но на практике, Windows с архитектурой x64 в настоящий момент поддерживает только 16 ТБ (это 2^44). Основное ограничение связано с тем, что современные процессоры могут обеспечивать доступ лишь к 1 терабайту физической памяти (это 2^40).

Как и в Windows с архитектурой x86, адресуемый диапазон памяти делится на пользовательские адреса и на системные. Каждый процесс получает 8 ТБ памяти и 8 ТБ остается для ядра ОС (напомню, что в Windows с архитектурой x86 было выделено 2 ГБ для процесса и 2ГБ для ядра ОС).

Размер страницы в памяти также составляет 4 КБ. Первые 64 КБ адресного пространства никогда не отображаются, т.е. наименьший правильный адрес это 0x10000. Также стоить знать, что в Windows с архитектурой x64 системные DLL загружаются в память по адресу, который всегда выше 4 ГБ.

Регистры процессора

Основным отличием архитектуры x64 от x86 является увеличение количества регистров и их размера.

Архитектура x64 имеет:

- 16 целочисленных 64-битных регистра общего назначения (RAX, RBX, RCX, RDX, RBP, RSI, RDI, RSP, R8 — R15);
- 8 регистров с плавающей точкой (FPR0 — FPR7);
- 8 регистров Multimedia Extensions (MM0 — MM7, имеют общее пространство с регистрами ST0 — ST7);

- 16 128-битных регистров SSE (XMM0 — XMM15);
- 64-битный указатель RIP
- 64-битный регистр флагов RFLAGS.

Передача аргументов в функцию

Как вы помните, в архитектуре x86 аргументы в функции передавались через стек. В архитектуре x64 аргументы в функции передаются немного по-другому. Первые 4 аргумента передаются через регистры RCX, RDX, R8, R9, а остальные через стек.

Для аргументов с плавающей точкой используются регистры XMM0-XMM3, а также стек.

Обзор PE-файла с архитектурой x64

В качестве примера была подготовлена программа prog-6.1. Давайте посмотрим на ее исходный код.

Исходный код программы prog-6.1c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(int argc, char* argv[]) {
char userInput[11];
char currentPassword[] = "1212121212";

memset(userInput, 0, sizeof(userInput));

printf("Insert key: ");
scanf_s("%s", userInput);

if (strlen(userInput) > sizeof(userInput)) {
printf("ERROR. License key should contains 10 symbols.\n");
system("pause");
return 1;
}

if (strcmp(userInput, currentPassword) != 0) {
printf("ERROR. Your license key is invalid.\n");
system("pause");
return 1;
}
```

```
printf("Congratulations! Your copy of application is licensed.\n");

system("pause");
return 0;
}
```

Давайте попробуем скомпилировать программу под архитектуру x64 и посмотрим на отличия в PE-заголовке.

```
> cl prog-6.1.c -Od /Gs- /GS- /link /DYNAMICBASE:NO /NXCOMPAT:NO
```

Теперь давайте посмотрим на PE-заголовок программы prog-6.1. Запускаем утилиту Puppy и открываем PE-файл prog-6.1.exe.

Выбираем NT Header

Видим, что - это PE-файл.

Выбираем File Header

Файл скомпилирован под архитектуру x64
В файле 5 секций

Выбираем Optional Header

Тип заголовка PE64

Это значит, что у нас PE-заголовок для 64-битных приложений. Заголовок PE64 отличается от обычного PE лишь тем, что в PE64 используются 64-битные виртуальные адреса.

Выбираем Section Headers

Здесь мы видим все 5 секций нашего PE-файла.

Если мы попробуем открыть программу prog-6.1 под отладчиком OllyDbg, то получим ошибку, так как OllyDbg умеет работать только с 32-битными PE-файлами.

В следующих видео этого урока мы познакомимся с инструментами, которые умеют работать с PE-файлами с архитектурой x64.